Database Systems

Lecture #2

Dr Sherin ElGokhy

Chapter 2: The relational model

Logical data models

- Hierarchical and network closer to physical structures, relational model provides a higher level of data independence
  - in the relational model we have only values: even references between data in different sets (relations) are represented by means of values.
  - The relational model satisfies the requirement of data independence.
- The hierarchical and network model include explicit references to the underlying structure, by means of the use of pointers and the physical ordering of data.

- More recently, the **object** model has been introduced

The relational model

- Proposed by **E. F. Codd in 1970** in order to support data independence.
- Made available in commercial DBMSs in 1981 (it is not easy to implement data independence efficiently and reliably!).
- It is based on (a variant of) the mathematical notion of **relation.**
- Relations are naturally represented by means of **tables**

Mathematical relations

- $D_1, D_2, \ldots, D_n$ (n sets, not necessarily distinct)
- **Cartesian product** $D_1 \times D_2 \times \ldots \times D_n$:
  - The set of all (ordered) n-tuples (d1, d2, …, dn) such that:

$$d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n$$

$D_1 \times D_2$ = set of ordered pairs (d_1, d_2); $d_1 \epsilon D_1$, $d_2 \epsilon D_2$

Ex:

Set A={1,2,4} and Set B={a,b}

Cartesian product is:

A×B={(1,a),(1,b),(2,a),(2,b),(4,a),(4,b)}

- ordered & not repeated; **(1,a) not (a,1)**
- No. of elements is 6 (3*2)

1	a
1	b
2	a
2	b
4	a
4	b

Tabular Form

Mathematical relations

- A **mathematical relation** on $D_1, D_2, \ldots, D_n$:
  - is a subset of the Cartesian product $D_1 \times D_2 \times \ldots \times D_n$.

Relation is a subset of the Cartesian product

Ex:

Relation={(1,a),(1,b),(4,b)} from the previous A×B

- $D_1, D_2, \ldots, D_n$ are the **domains of the relation** → **n-sets**
- n is the **degree** of the relation.

Degree → No. of columns (attributes)

- Each pair is called **tuple.**
- The number of n-tuples is the **cardinality** of the relation; in practice, it is always finite.

Cardinality → No. of rows (tuples)

Relations and tables

- **Example:**
  - No. of sets (domains)=3

    {x,y} {a,b,c} {3,5}
  - For the Cartesian product:
    - Degree = 3
    - Cardinality = 12
  - For the Relation:
    - Degree = 3
    - Cardinality = 6

x	a	3
x	a	5
x	c	5
y	a	3
y	c	3
y	c	5

x	a	3
x	a	5
x	b	3
x	b	5
x	c	3
x	c	5
y	a	3
y	a	5
y	b	3
y	b	5
y	c	3
y	c	5

Cartesian product in a Tabular Form

Relations with tables

- A mathematical relation is a **set** of **ordered** n-tuples

  $(d_1, d_2, \ldots, d_n)$ with $d_1 \in D_1$, $d_2 \in D_2$, …, $d_n \in D_n$

- A set, so:
  - there is no ordering between n-tuples.
  - the n-tuples are distinct from one from the other.

- The n-tuples are **ordered**: the i^{th} value come from the i^{th} domain: so there is an ordering among the domains

Relations with tables

Real Madrid	Roma	3	1
Liverpool	Milan	2	0
Real Madrid	Liverpool	1	2
Roma	Milan	0	1

A relation with the results of soccer matches

- In a mathematical relation we have n-tuples whose elements are distinguished by position.
- A non-positional notation is introduced, by associating names with the domains in a relation, referred to as attributes, which describe the 'roles' played by the domains.
- Given the necessity of identifying the components unambiguously, the attributes of a relation (and therefore the column headings) must be different from each other.

Relations with tables

Non-positional technique: the elements are distinguished by the attributes' names (column headings)

HomeTeam	VisitingTeam	HomeGoals	VisitingGoals
Real Madrid	Roma	3	1
Liverpool	Milan	2	0
Real Madrid	Liverpool	1	2
Roma	Milan	0	1

A relation with the attributes

Relations and databases

- **Database** made up of several **relations**.
- **Example:**
  - STUDENT (RegNum, Surname, FirstName, BirthDate)
  - COURSE (Code, Title, Tutor)
  - EXAMS (Student, Grade, Course)

STUDENTS

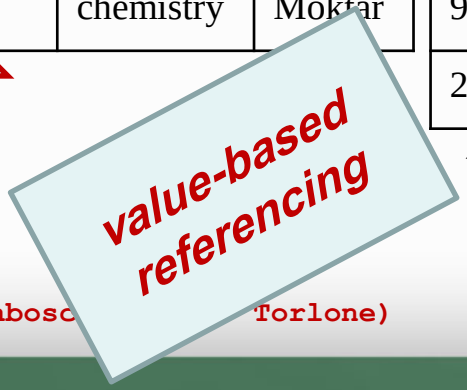
RegNum	Surname	FirstName	BirthDate
276545	Fawzy	Moner	2/11/1992
485745	Kamel	Ahmed	23/6/1990
200768	Selim	Karim	1/5/1992
587611	Moaz	Sayed	23/8/1991
937653	Maher	Maher	5/10/1990

COURSES

Code	Title	Tutor
01	Physics	Hatem
03	chemistry	Ali
04	chemistry	Moktar

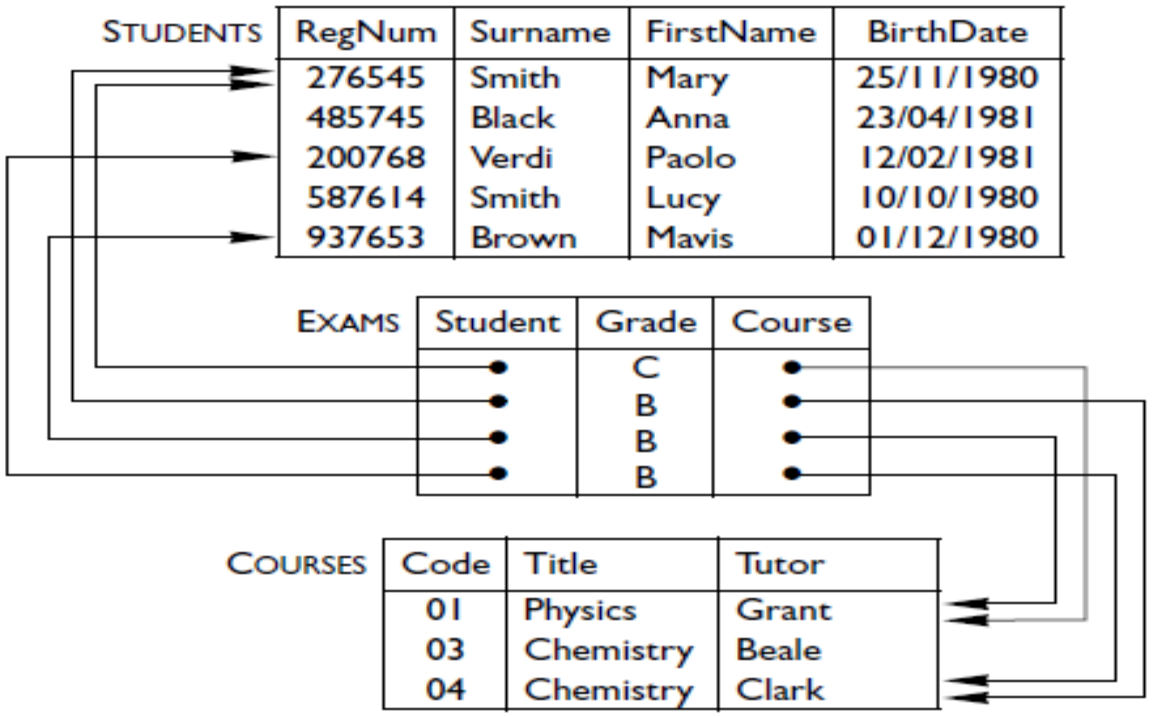
EXAMS

Student	Grade	Course
276545	C	01
276545	B	04
937653	B	01
200768	B	04



Relations and databases

• The network and hierarchical models represent references explicitly by means of pointers and for this reason are called 'pointer-based' models.



STUDENTS

RegNum	Surname	FirstName	BirthDate
276545	Smith	Mary	25/11/1980
485745	Black	Anna	23/04/1981
200768	Verdi	Paolo	12/02/1981
587614	Smith	Lucy	10/10/1980
937653	Brown	Mavis	01/12/1980

EXAMS

Student	Grade	Course
	C	
	B	
	B	
	B	

COURSES

Code	Title	Tutor
01	Physics	Grant
03	Chemistry	Beale
04	Chemistry	Clark

A database with pointers.

Incomplete Information and NULL values

- Relational model is simple and effective technique **but it is rigid.**
  - **DBMS do not accept empty values or 0.**
  - **Null value**: a special value (not a value of the domain) denotes the absence of a domain value (place holder).

Example:

City	Governorate_Address
Cairo	Kasr El-Ainy Street
Alex	NULL
Damanhour	NULL
Luxor	NULL

Unknown value

Non-existent value

No-information value

Incomplete Information and NULL values (cont.)

- **Types of Null Value:**
  - **Unknown value**: there is a domain value, but it is not known (Alex)
  - **Non-existent value**: the attribute is not applicable for the tuple (Damanhour)
  - **No-information value**: we don‘t know whether a value exists or not (Luxor); this is the disjunction (logical or) of the two preceding.

Incomplete Information and NULL values (cont.)

- **DBMSs** do not distinguish between the types: they implicitly adopt the no-information value.
- It is possible for tuples to have a null value, denoted by ***Null***, for some of their attributes.
- ***Null*** signifies an unknown value or that a value does not exist.
- The result of any arithmetic expression involving ***null* is *null.***

Incomplete Information and NULL values (cont.)

- **Example:**

STUDENTS

RegNum	Surname	FirstName	BirthDate
276545	Fawzy	Moner	2/11/1992
485745	Kamel	Ahmed	23/6/1990
200768	Selim	Karim	1/5/1992
587611	Moaz	Sayed	23/8/1991
937653	Radwan	Maher	5/10/1990

EXAMS

Student	Grade	Course
276545	C	01
276545	B	04
937653	B	01
200768	A	04

COURSES

Code	Title	Tutor
01	Physics	Hatem
03	chemistry	Ali
04	chemistry	Moktar

Incomplete Information and NULL values (cont.)

- **Example:**

STUDENTS

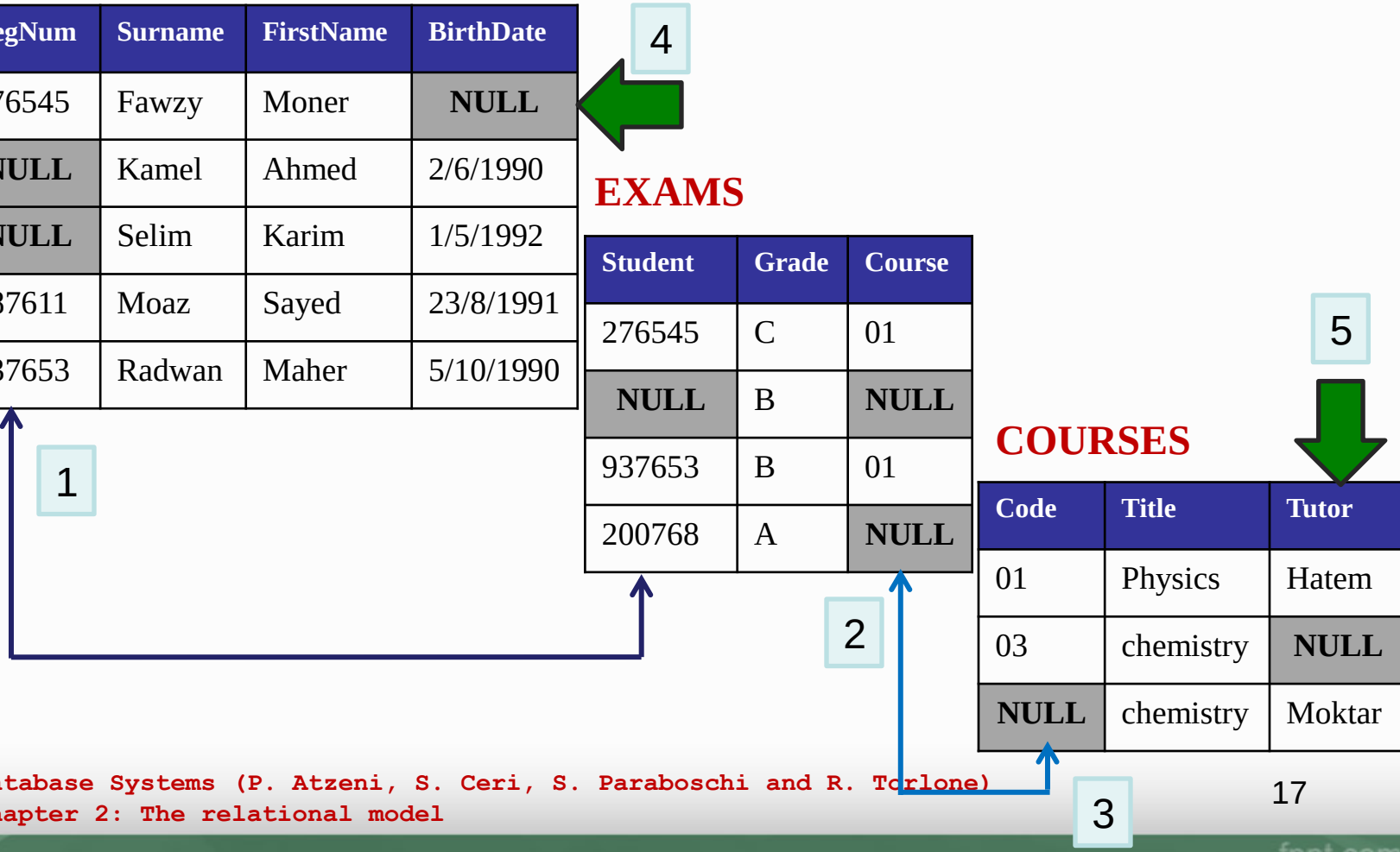
RegNum	Surname	FirstName	BirthDate
276545	Fawzy	Moner	**NULL**
NULL	Kamel	Ahmed	2/6/1990
NULL	Selim	Karim	1/5/1992
587611	Moaz	Sayed	23/8/1991
937653	Radwan	Maher	5/10/1990

EXAMS

Student	Grade	Course
276545	C	01
NULL	B	**NULL**
937653	B	01
200768	A	**NULL**

COURSES

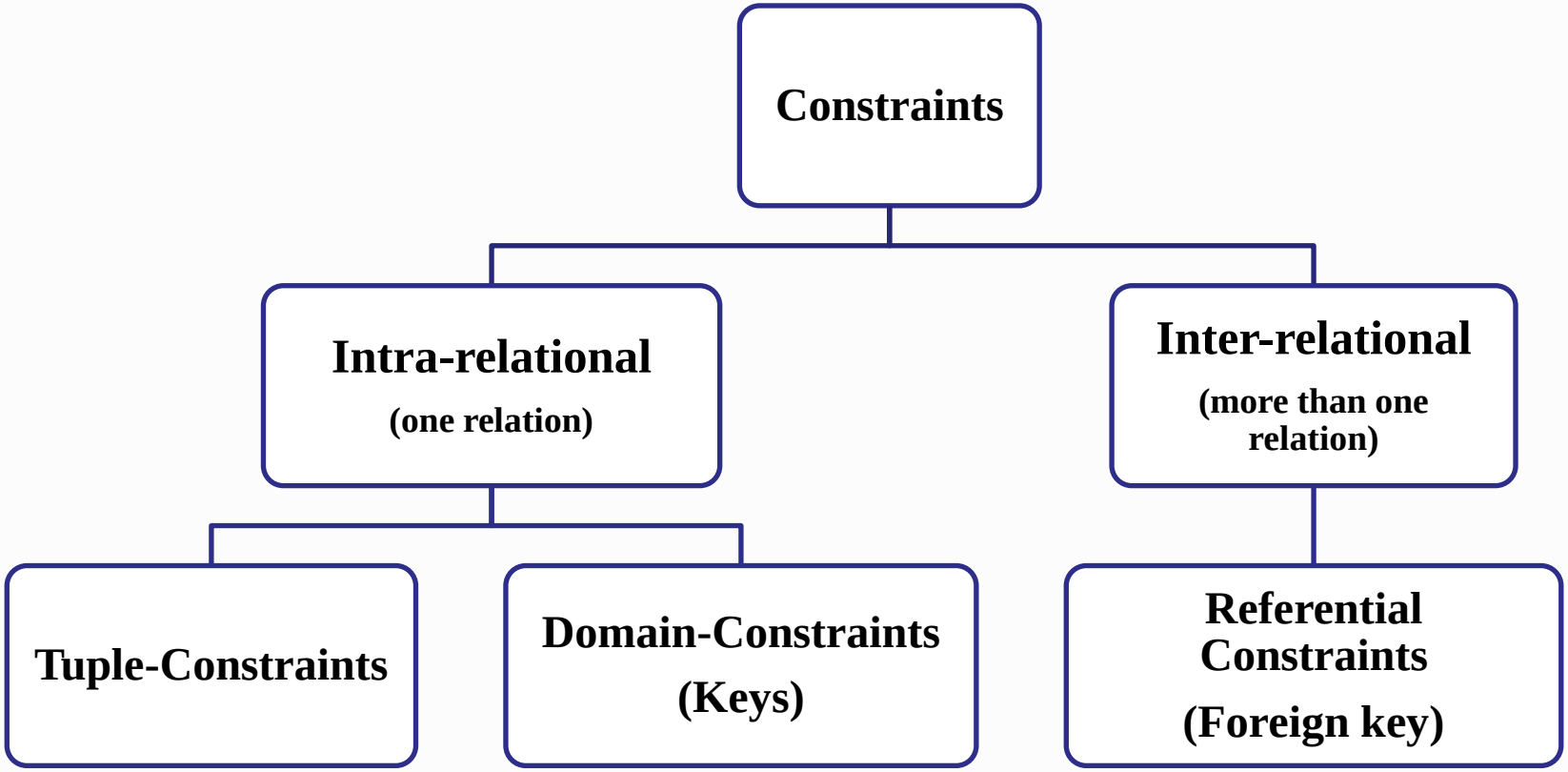
Code	Title	Tutor
01	Physics	Hatem
03	chemistry	**NULL**
NULL	chemistry	Moktar



Integrity Constraints

- **Integrity constraints**: a property that must be satisfied by all correct database instances;
- A database instance is **legal** if it satisfies all integrity constraints
- types of constraints
  - ***Intra-relational constraints***; special cases:
    - tuple constraints
    - domain constraints (Keys)
  - ***Inter-relational*** constraints
    - Referential constraints

Integrity Constraints



Integrity Constraints

- **Tuple Constraints:**
  - express conditions on the values of each tuple, independently of other tuples
  - a possible syntax: boolean expressions with atoms that compare attributes, constants or expressions over them
- **Domain Constraint**: a tuple constraint that involve a **single attribute**

Examples:

- A domain constraint

  (**Grade** ≥ "A") AND (**Grade** ≤ "F")

- A tuple constraint

  (NOT (**Honours** = "honours"))OR (**Grade** = "A")

- A tuple constraint (on another schema) with expressions:

  Net = Amount - Deductions

Integrity Constraints

- **Example:**

STUDENTS

RegNum	Surname	FirstName	BirthDate
276545	Selim	Karim	12/02/1991
937653	Abdel-Aziz	Sayed	10/10/1992
937653	Radwan	Maher	01/12/1990

1

EXAMS

Student	Grade	Honors	Course
200768	K	NULL	05
937653	B	honors	01
937653	A	honors	04
276545	G	NULL	01

4 2 3 5

COURSES

Code	Title	Tutor
01	Physics	Hatem
03	chemistry	Ali
04	chemistry	Moktar

Integrity Constraints

1. The last two tuples of the relation STUDENTS contain information on two different students with the same registration number: again an impossible situation, given that the registration number exists for the precise purpose of identifying each student unambiguously.
2. In the first tuple of the relation EXAMS we have an exam result of K, which is not admissible, as grades must be between A and F.
3. In the second tuple again in the relation EXAMS an honours is shown awarded for an exam for which the grade is B. Honours can be awarded only if the grade is A.

Integrity Constraints

4. The first tuple of the relation EXAMS shows, for the attribute Student, a value that does not appear among the registration numbers of the relation STUDENTS: this is also an unacceptable situation, given that the registration number provides us with information only as a link to the corresponding tuple of the relation STUDENTS.

5. Similarly, the first tuple shows a course code that does not appear in the relation COURSES.

Integrity Constraints

- **Keys Constraints:**
- Key:
  - A set of attributes that uniquely identifies tuples in a relation
- More precisely:
  - **A Superkey:** a set of attributes K is a **superkey** for a relation r if r does **not** contain two distinct tuples t_1 and t_2 with $t_1[K]=t_2[K]$;
  - **A key:** K is a **key** for r if K is a minimal superkey (that is, there exists no other superkey K' of r that is contained in K as proper subset)

    Key = Minimal SuperKey
  - Key by chance.
  - Keys and NULL values

Integrity Constraints

- **Example:**

Key

RegNum	Surname	FirstName	BirthDate	DegreeProg
284328	Abdel-Aziz	Fatehy	29/04/1989	Computing
296328	Abdel-Aziz	Rashed	29/04/1989	Computing
587614	Abdel-Aziz	Sayed	01/05/1999	Engineering
934856	Kamel	Sayed	01/05/1999	Fine Art
965536	Kamel	Sayed	05/03/1988	Fine Art

- (**RegNum**, DegreeProg) → SuperKey
- (Surname, FirstName, BirthDate) → SuperKey
- (FirstName, DegreeProg) → Not a SuperKey so not a Ke
- If the values of (DegreeProg) are different
  - we can called (FirstName, DegreeProg) → Key by Chance

Primary Key: is a **key** that does not contain **NULL** value or **repeated** values

Integrity Constraints

Referential Constraints (Foreign Key)

- Pieces of data in different relations are correlated by means of values of (primary) keys
- **A referential constraints** imposes to the values on a set X of attributes of a relation $\mathbf{R_1}$ to appear as values for the **primary key** of another relation $\mathbf{R_2}$**.**
- The referential constraint between the attribute A of $\mathbf{R_1}$ and the relation $\mathbf{R_2}$ is satisfied if, for every tuple t_1 in $\mathbf{R_1}$ such that $t_1[A]$ is not null, there exists a tuple $\mathbf{t_2}$ in $\mathbf{R_2}$ such that

$$\mathbf{t_1}[A] = \mathbf{t_2}[\text{Primary key attribute of } \mathbf{R_2}].$$

- In the more general case, we must take account of the fact that each of the attributes in X must correspond to a precise attribute of the primary key K of $\mathbf{R_2}$.

Primary & Foreign Keys

- **Example:**

STUDENTS

RegNum	Surname	FirstName	BirthDate
276545	Fawzy	Moner	25/11/1992
485745	Kamel	Ahmed	23/6/1990
200768	Selim	Karim	1/5/1992
587611	Abdel-Aziz	Sayed	3/8/1991

Referential constraint (Foreign Key)

EXAMS

Student	Course	Grade
276545	01	C
276545	04	B
200768	01	B
200768	04	A

Referential constraint (Foreign Key)

COURSES

Code	Title	Tutor
01	Physics	Hatem
03	chemistry	Ali
04	chemistry	Moktar

Thanks